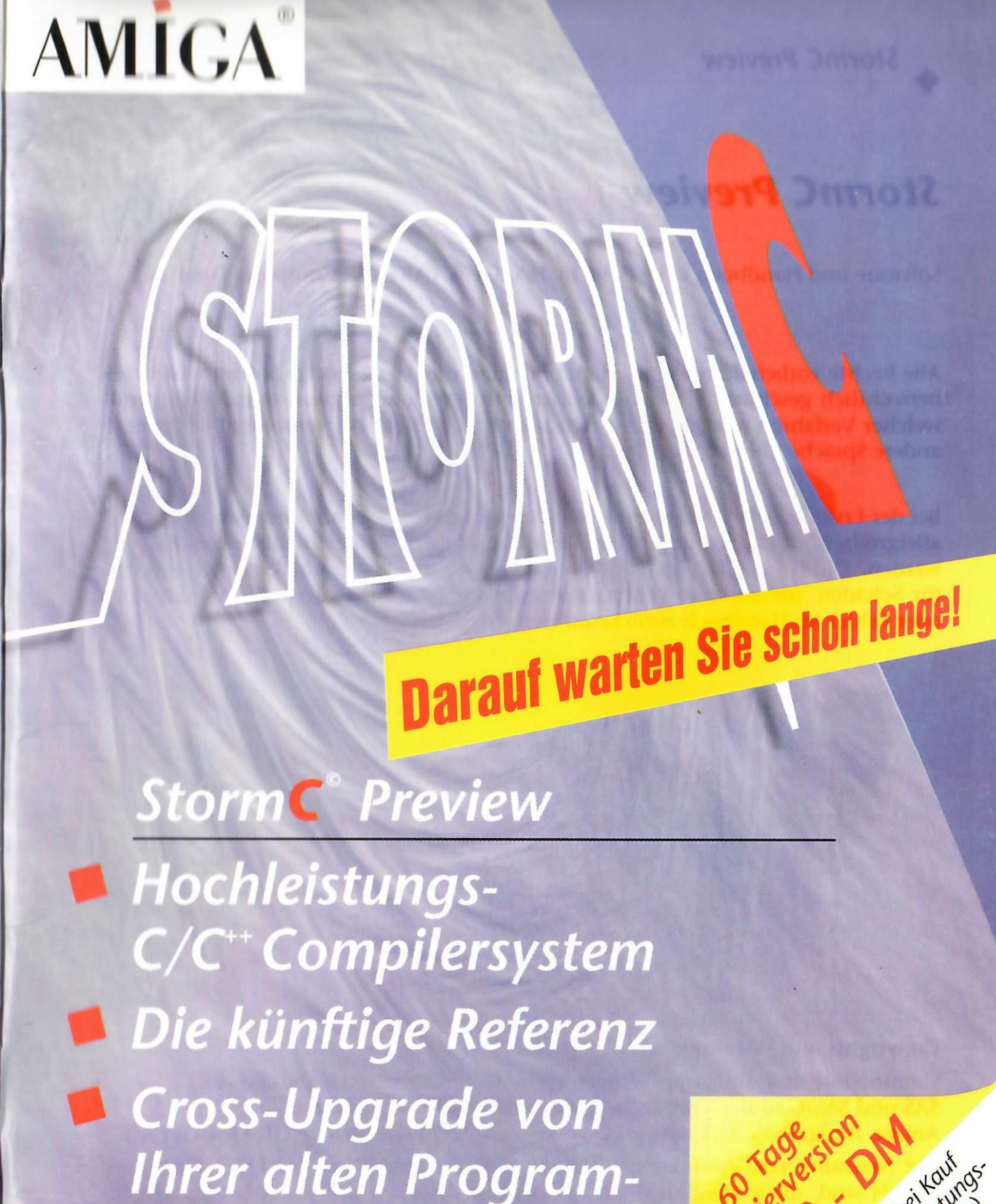


StormC© Preview

- Hochleistungs-C/C++ Compilersystem
- Die künftige Referenz
- Cross-Upgrade von Ihrer alten Programmiersprache möglich

60 Tage Probierversion nur 20,- DM

(Schutzgebühr wird bei Kauf angerechnet • Gewährleistungsausschluß siehe Rückseite)

StormC Preview

Bei der Erstellung des Programms, der Anleitung sowie der Abbildungen wurde mit allergrößter Sorgfalt vorgegangen. Trotzdem können Fehler nicht ausgeschlossen werden. Die HAAGE & PARTNER Computer GmbH übernimmt keinerlei Haftung für Schäden, die auf eine Fehlfunktion des Programms, fehlerhafte Abbildungen oder Fehler im Handbuch zurückzuführen sind.



Inhaltsverzeichnis



Willkommen zu einer neuen Ära der Amiga-Programmierung.

compile – engl.: Zusammenstellen, Zusammentragen

Programm, das Befehle (Quelltext) einer Hochsprache in Maschinensprache (Objektcode) übersetzt.

Mit der beiliegenden Vorschau unseres brandneuen Compilersystems lernen Sie die Fähigkeiten einer fortschrittlichen Programmiersprache kennen.

In einer sogenannten integrierten Umgebung finden Sie alles, was Sie zum Programmieren benötigen.

Herzstück und Steuerzentrale ist die Projektverwaltung, von der aus alle weiteren Komponenten des Systems aufgerufen und mit Daten versorgt werden.

Die Projektverwaltung ist nicht einfach nur ein besseres Make, sondern eine Verwaltung für alle Ihre Programmodule. Dazu zählen nicht nur Quelltexte und Objekt-Bibliotheken, auch die Programmdokumentation, zum Programm gehörende Arexx-Scripts, Bilder und Ressourcen werden darin verwaltet. Selbstverständlich werden von hier aus auch alle Compiler-, Editor- und Projekt-Optionen eingestellt.

Make – engl.: machen, anfertigen, hervorbringen.

Bei Compilersystemen veranlaßt Make, daß die tatsächlich veränderten Programmodule kompiliert werden.

Wenn Sie jetzt der Meinung sind, daß das alles bestimmt viel zu kompliziert zu bedienen ist, kann ich Sie beruhigen. Blättern Sie doch einmal auf die nächsten Seiten, dorthin, wo das erste Beispiel beschrieben wird. Anhand der Bilder können Sie bereits sehen, daß alles sehr einfach und intuitiv bedient werden kann.

Syntax

Formale Regeln, die in einer Programmiersprache eingehalten werden müssen.

Eine weitere Komponenten im System ist z. B. der Editor mit seiner besonderen Fähigkeit, Schlüsselworte und Syntax-Eigenschaften farbig im Text hervorzuheben. Die Farbmarkierung hilft nicht nur, daß Sie ihr Programm durch die besondere Strukturierung besser lesen können, sie hilft auch dabei, Fehler bei der Erfassung Ihrer Quelltexte zu vermeiden. Sobald ein Schlüsselwort oder eine Amiga-Funktion komplett eingegeben ist, wird das Wort farbig markiert und Sie haben die Sicherheit, daß die Schreibweise korrekt ist.

Bevor wir zum eigentlichen Arbeitspferd, dem Compiler kommen, möchte ich kurz auf den außerge-



wöhnlichen Debugger eingehen. Außergewöhnlich deshalb, weil Sie den Unterschied, ob nun der Editor läuft oder ob es doch der Debugger ist, kaum bemerken werden. Der Debugger nutzt nämlich die Fähigkeiten des Editors. Die Quelltexte des zu debuggenden Programmes werden im Editorfenster angezeigt.

Debugger – engl.: Entwanzer

Programm zum Suchen und Entfernen von Programmierfehlern ("Bugs").

Wie gewohnt können Sie nun darin blättern, Unterbrechungspunkte setzen und Programmfunktionen oder Variablen suchen. Wie gewohnt wird auch die Quelltext-Strukturierung durch die farbige Textdarstellung unterstützt. Sie können so wesentlich flüssiger arbeiten.

Wir hatten natürlich bei der bisherigen Planung auch einen Hintergedanken. Für zukünftige Versionen ist es vorgesehen, daß Änderungen, die man während des Debuggens in den Quelltext einfügt, direkt wieder übersetzt werden können. Das lästige Beenden des Debugger, das Neukompilieren und erneute Starten hat dann ein Ende. Schöne Aussichten, die die Software-Entwicklung auf dem Amiga noch effizienter gestalten.

Eine sehr große Hilfe für den Debugger ist auch unsere RunShell. Durch sie ist es möglich, die typischen Programmierfehler bei der Betriebssystemprogrammierung schnell zu lokalisieren.

Exception – engl.: Ausnahme

In dem beschrieben Fall betrifft dies die CPU, die in den Ausnahmezustand versetzt wird, was im Normalfall (wenn es nicht abgefangen wird) in einem GURU (Systemabsturz) endet!

Ein Beispiel für Fehler, die immer wieder gemacht werden, ist am besten mit den Funktionen `AllocMem()` und `FreeMem()` zu beschreiben: Speicher zu allozieren, ist eine einfache Sache, ihn aber wieder an das System zurückzugeben, stellt für viele Programmierer ein großes Problem dar. Entweder wird vergessen, alle Speicherblöcke freizugeben oder der freigegebene Block ist zu groß oder zu klein, was meist in einer CPU-Exception endet. Die RunShell führt zu allen wichtigen Systemroutinen Protokoll, die etwas mit den Systemressourcen zu tun haben. So kann genau dokumentiert werden, wann Funktionen zuwenig, zuoft oder mit fehlerhaften Parametern aufgerufen werden. Ein weiterer großer Vorteil der RunShell ist die Möglichkeit, jederzeit während des Programmablaufs den Debugger zu starten. Sie müssen sich also nicht bereits vor dem Programmstart entscheiden, ob Sie nun debuggen möchten,

oder ob das Programm im normalen Modus ablaufen soll. Wechseln Sie einfach im Bedarfsfall in den Debugger.

Aber nun zum eigentlichen Gehirn des Entwicklungspaketes, dem ANSI C/C++ Compiler.

Der Begriff objektorientierte Programmierung ist in aller Munde. Kaum ein Softwarehaus entwickelt heute noch in ANSI C. Zumindest wird dieser Eindruck vermittelt. Das Gegenteil ist jedoch der Fall. Alle benutzen zwar einen C++ Compiler, aber der ist natürlich genauso gut geeignet, auch ANSI C Quelltexte zu übersetzen.

Unser Augenmerk richtet sich deshalb auch an diejenigen, die bisher mit ANSI-C gearbeitet haben, und bei einem Umstieg auf ein zukunftsicheres Compilersystem ihre Programme 1 zu 1 weiterverwenden möchten. Der sanfte Umstieg ist mit StormC gewährleistet.

Die C++-Implementation richtet sich nach den Vorgaben von Bjarne Stroustrup und ist damit zum erweiterten AT&T Standard 3 kompatibel.

Der Compiler generiert Code für alle Motorola 680x0 CPUs einschließlich der 68060 CPU, die immer mehr auf Turboboards zum Einsatz kommt. Die grundsätzlich überragende Kompiliergeschwindigkeit wird durch den Einsatz von vorkompilierten Headerdateien (precompiled Header) um Faktoren beschleunigt. Der integrierte Linker verarbeitet alle gängigen Bibliotheksformate (Standardformate, SAS/C, MaxonC++, ...) und ist dabei einer der schnellsten auf dem Amiga.

StormC ist für alle Entwicklungsaufgaben geeignet, seien es Verwaltungs-, Grafik-, Musik- oder Spiel-Programme. Auch die Betriebssystem-Entwicklung ist für StormC kein Problem.

Die vorliegende Preview-Version von StormC soll Ihnen bei der Entscheidung für Ihr zukünftiges Compilersystem helfen. Eine selbstablaufende



Demoversion wäre dabei aber mehr als unsinnig. Schließlich wollen Sie auch eigene Projekte ausprobieren und eventuell testen, wie sich das System gegenüber Ihrem alten verhält. Die Preview-Version von StormC bietet Ihnen daher die Möglichkeit, alle Funktionen des Compilers und der integrierten Umgebung genau unter die Lupe zu nehmen.

Bitte bedenken Sie allerdings, daß es sich bei der vorliegenden Preview-Version nicht und das fertige Produkt handelt. In der ersten Version, die im Januar ´96 erscheinen wird, sind noch etliche Funktionen mehr enthalten und die Funktionsweise des Gesamtsystems noch harmonischer. Sollten eventuell einzelne Funktionen nicht so arbeiten, wie Sie es sich wünschen, oder haben Sie sonstige Verbesserungsvorschläge, scheuen Sie sich bitte nicht uns anzurufen oder uns zu faxen. Wir werden jeden Wunsch entgegen nehmen und mit unserem Programmiererteam darüber beraten.

Philosophisches

Eine Programmiersprache ist für den Programmierer ein Hilfsmittel, um auszuführende Aktionen zu spezifizieren. Sie stellt eine Menge von Begriffen zur Verfügung, mit denen der Programmierer die Möglichkeiten der Problemlösung bedenken kann. Aber nicht nur die Sprache und ihre sprachlichen Hilfsmittel sind für den Programmierer wichtig, auch die Implementation der zugehörigen Programmier-Hilfsmittel auf der jeweiligen Hardware-Plattform spielen eine entscheidende Rolle für den Erfolg.

Ein Aspekt bei der Sprachwahl erfordert daher eine Sprache, die im Idealfall nah an der Maschine ist, so daß mit allen wichtigen Aspekten der Maschine, für den Programmierer transparent und nachvollziehbar, handlich und effizient umgegangen werden kann. C ist eine Sprache, die primär unter diesem Aspekt entworfen wurde.

Der zweite Aspekt erfordert eine Sprache, die nah am Problem ist, so daß die Konzepte der Problemlö-

sung direkt und schlüssig formuliert werden können. Die Sprachelemente, die in C⁺⁺ über C hinausgehen, wurden in Hinblick auf diesen zweiten Aspekt entworfen.

Der dritte Aspekt betrifft die Implementation des Compilersystems und seiner Hilfsmittel. Die Programmierumgebung sollte dem Programmierer die Möglichkeit geben, sich ganz auf die Problemlösung zu konzentrieren. Die einfache und logisch durchdachte Handhabbarkeit vom StormC erfüllt diese Anforderung.

Anforderungen

In der folgenden Liste finden Sie die Minimal-Konfiguration zum Betrieb von StormC:

- Amiga mit Festplattenlaufwerk
- Kickstart/Workbench 2.0 (v37)
- 3 MB Hauptspeicher
- 5 MB Festplattenspeicher

Mit dieser Zusammenstellung kann zwar entwickelt werden, aber die Projektgröße ist sehr begrenzt. Desweiteren können nicht alle Debugger-Eigenschaften genutzt werden, wenn nicht wenigstens eine 68030 CPU mit MMU eingesetzt wird.

Am besten eignet sich die folgende Konfiguration:

- Amiga mit mind. 68030 inkl. MMU
- Kickstart/Workbench 3.0 besser 3.1
- 8 MB freier Hauptspeicher
- 30 MB freier Festplattenspeicher

In dieser Konfiguration werden der Compiler und auch die Shell kaum über zu wenig Speicher klagen.

Als Faustformel können Sie sich jedoch merken: VIEL HILFT VIEL!

Installation

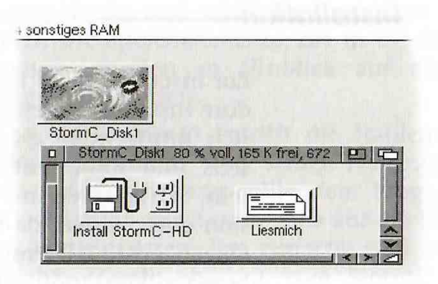
Zur Installation auf Ihre Festplatte wird der Commodore Installer eingesetzt. Dieses Installationstool hat sich mittlerweile als Standard-Installierer durchgesetzt und sollte Ihnen in der Bedienung bekannt sein. Wir mußten uns für die englischsprachige Version entscheiden, da beim Einsatz der deutschsprachigen eventuell Probleme auftreten können. Zum besseren Verständnis finden Sie hier die Übersetzung der wichtigsten Tasten:

Proceed	OK und weiter. Die abgefragte Aktion wird ausgeführt.
Abort Installation	Installation beenden. Die Installation wird nicht fortgeführt.
Parent Drawer	Mutterverzeichnis Den Inhalt des nächst höheren Verzeichnisses anzeigen.
Show Drives	Laufwerke Alle Laufwerke anzeigen.
Make New Drawer...	Erzeuge neues Verzeichnis. Legt einen neuen Ordner im angegebenen Pfad an.
Help...	Das sollten Sie auch ohne Übersetzung wissen!
Cancel	Aktion abbrechen.

Legen Sie bitte die erste Diskette der Preview-Version in Ihr Diskettenlaufwerk und Doppelklicken Sie auf das Disketten-Ikon.



Bevor Sie nun mit der Installation beginnen, sollten Sie sich die Datei "**Liesmich**" auf der Diskette durchlesen. Hierin sind wichtige Neuerungen und Tips beschrieben, die leider nicht mehr mit ins Handbuch übernommen werden konnten.



Script – engl.: Schriftstück, Manuskript

In unserem Fall handelt es sich um den interpretierbaren Installations-Quelltext zum Installieren von StormC.

Doppelklicken Sie jetzt auf das Ikon "**Install StormC-HD**". Bitte haben Sie einen Augenblick Geduld, bis das Installationsprogramm und das -Script geladen sind.



Bitte folgen Sie den Anweisungen des Installationsprogrammes. Sollten Sie einmal nicht weiter wissen, klicken Sie einfach auf den "Help"-Knopf und lesen Sie nach, was zu tun ist.

Nach erfolgreicher Installation erhalten Sie eine entsprechende Meldung vom Installationsprogramm.

Sollte die Installation nicht positiv verlaufen sein, wiederholen Sie bitte den Vorgang mit eingeschalteter "Log-Datei"-Generierung. Die Option "Log all actions to: Log File" kann in dem Optionen-Fenster eingestellt werden, das nach dem Begrüßungs-Fen-



ster angezeigt wird. Nach der erfolglosen Installation können Sie dann im Installationsprotokoll nachlesen, was nicht funktioniert hat. Beheben Sie bitte das Problem und installieren Sie erneut.

Was tun bei „unlösbaren" Problemen

Sollten Sie nicht nur bei der Installation Schwierigkeiten mit der Software haben, scheuen Sie sich bitte nicht, uns anzurufen. Wir sind gerne bereit, Ihnen zu helfen und gemeinsam mit Ihnen die Probleme aus der Welt zu schaffen.

Hier können Sie uns erreichen:

Werktags zwischen 9:00 und 17:00 Uhr unter:

06007/930051

Per E-Mail:

Compuserve: 100654,3133

Internet: 100654.3133@compuserve.com

Tutorial

Im Tutorialteil des Handbuchs erfahren Sie alles über den Umgang mit dem Programm. Anhand eindrucksvoller Beispiel wird Ihnen die Funktionsweise des Compilersystems beigebracht. Sie lernen den Umgang mit der Projektverwaltung, wie man Quelltexte erfaßt und schließlich, wie man den Compiler startet. In ein weiteres Beispiel beschreibt den Umgang mit dem Debugger. Ein kleines Demoprogramm wird erstellt und Schritt für Schritt im Debugger abgearbeitet.

Sie erhalten durch das Tutorial einen umfassenden Eindruck des Compilersystems und werden, nach dem Sie es durchgearbeitet haben, nie mehr mit einem anderen Compilersystem arbeiten wollen.

Programmstart

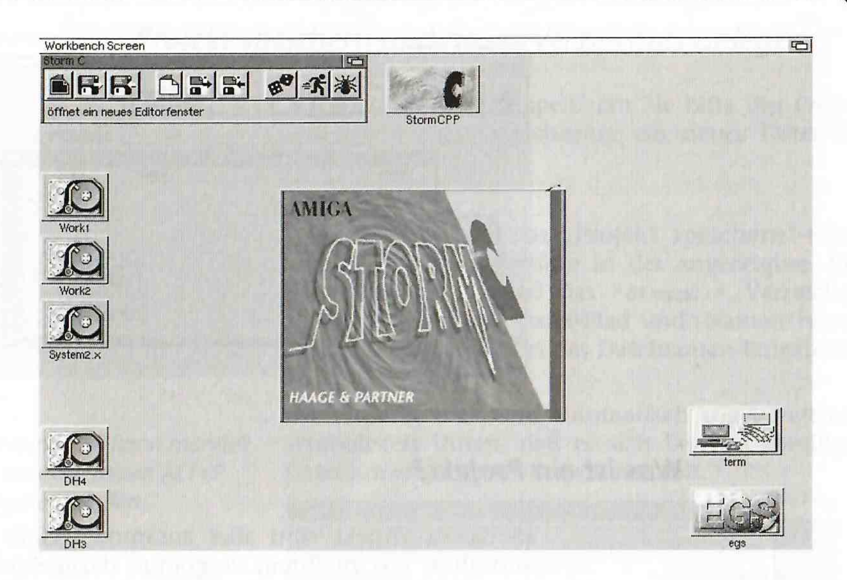
Im ersten Beispiel lernen Sie, wie Sie ein neues Projekt erstellen, den Programm-Quelltext erfassen und das Programm kompilieren und starten.



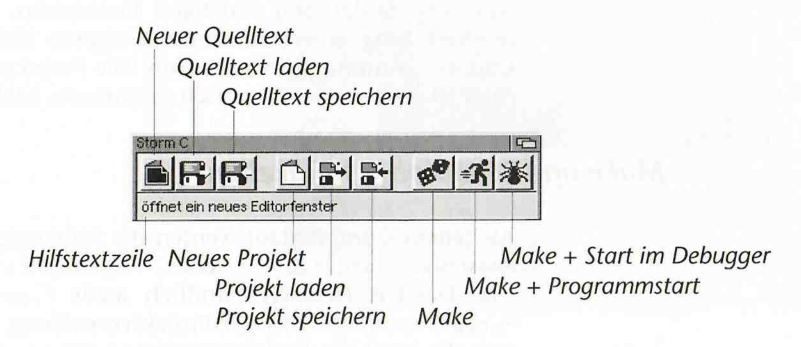
Starten Sie bitte StormC durch Doppelklick auf das Programm-Ikon (siehe nebenstehende Abbildung). Das Programm finden Sie in der Schublade, in das Sie das Compilersystem installiert haben.

Während des Starts sehen Sie eine Willkommensmeldung, die solange auf dem Bildschirm zu sehen ist, bis alle Programmkomponenten geladen sind.

Nach dem Laden aller Komponenten wird am oberen Bildschirmrand eine Piktogrammleiste angezeigt, die die wichtigsten Funktionen der integrierten Oberfläche zum schnellen Anklicken bereithält.



Die Piktogrammleiste stellt folgende Funktionen bereit:

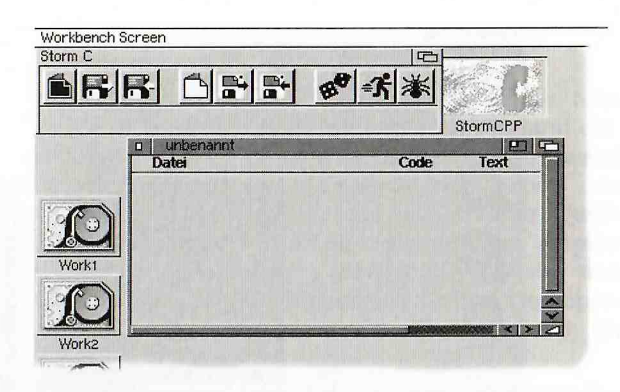


Wir beginnen selbstverständlich mit dem typischen Beispiel beim ersten Kontakt mit einem neuen Programmiersystem: `"HELLO WORLD"`.

Erzeugen eines neuen Projektes



Klicken Sie bitte in der Piktogrammleiste auf das Ikon "**Neues Projekt**". Ein neues Projektfenster wird angezeigt.



Was ist ein Projekt?

In einem Projekt wird alles zusammengefaßt und verwaltet, was zu Ihrem Programm dazugehört: C-, C++-, und Assembler-Quelltexte, Headertexte, Objekt-Dateien, Linker-Bibliotheken, Dokumentationen, Grafiken, Bilder und sonstigen Ressourcen. Durch die Trennung in verschiedene Sektionen bleibt die Dateien-Sammlung übersichtlich. Die Projektverwaltung ist aber auch ein grafisch orientiertes Make.

Make und Modulabhängigkeiten

Bei jedem Compilerlauf werden die Abhängigkeiten zwischen `".o"`-, `".h"`-, `".ass"`-, `".asm"`-, `".i"`- und `".c"`-Dateien (selbstverständlich auch `".cc"` oder `".cpp"`) ermittelt und der Projektverwaltung mitgeteilt. So weiß die Projektverwaltung ganz von alleine, daß ein C-Quelltext neu kompiliert werden muß, wenn eine `".h"`-Header-Datei verändert wurde, die im `".c"`-Quelltext „includet" wird.

Bei Klick auf die Piktogramme **„Make"** oder **„Run"** werden zuerst alle Abhängigkeiten geprüft und dann von Make entschieden, welche Programmodule neu kompiliert werden müssen und welche aktuell sind. **„Run"** unterscheidet sich gegenüber **„Make"** lediglich dadurch, daß bei erfolgreicher Kompilation das Programm automatisch gestartet wird



Projekt speichern und neues Verzeichnis anlegen

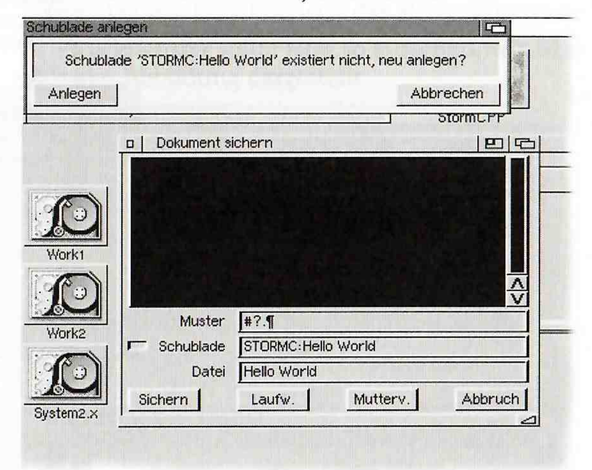
Als nächsten Schritt speichern Sie bitte das Projekt und legen dabei gleichzeitig ein neues Datei-Verzeichnis an.



Klick Sie bitte auf das „**Projekt speichern**"-Piktogramm. Wählen Sie bitte in der angezeigten ASL-Standard-Dateiauswahl das `"StormC:"` Verzeichnis aus und geben den Datei-Pfad und -Namen `"Hello World/Hello World"` in das Dateinamen-Eingabefeld ein.

Die Endung ".¶" kann manuell auch über die Tasten ALT+P eingegeben werden.

Die Endung `".¶"` wird automatisch angehängt und symbolisiert Ihnen, daß es sich bei der jeweiligen Datei um ein StormC-Projekt handelt.



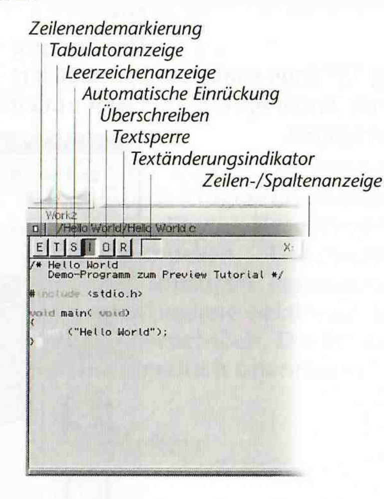
Sie werden sich sicher wundern, warum das leere Projekt gespeichert werden soll; auch wenn das Projekt noch keinen Inhalt hat, ist es sinnvoll bereits zu Beginn einen eindeutigen Projektpfad anzugeben. Die Dateinamen der Quelltexte und andere Ressourcen können dann relativ zum Projektpfad im Projekt aufgenommen und gespeichert werden. Andernfalls würde der absolute Pfad verwendet. Ein weiterer Vorteil ist, daß beim Hinzufügen von Projektdateien

bereits der entsprechende Pfad im ASL-Dialog eingetragen ist. Sie sparen sich dadurch die Pfadsuche.

Quelltext erfassen

Jetzt kommen wir zu der eigentlichen Programmierung. Öffnen Sie jetzt ein neues Quelltext-Fenster mit Klick auf das Piktogramm "**Neuer Quelltext**".

Bevor Sie nun anfangen Text einzugeben machen Sie sich bitte kurz mit den Bedienelementen des Editors vertraut.



Tippen Sie jetzt bitte folgende Programmzeilen ein:

```
/* Hello World
   Demo-Programm zum Preview-Tutorial */

#include <stdio.h>

void main( void)
{

printf("Hello World\n");
}
```



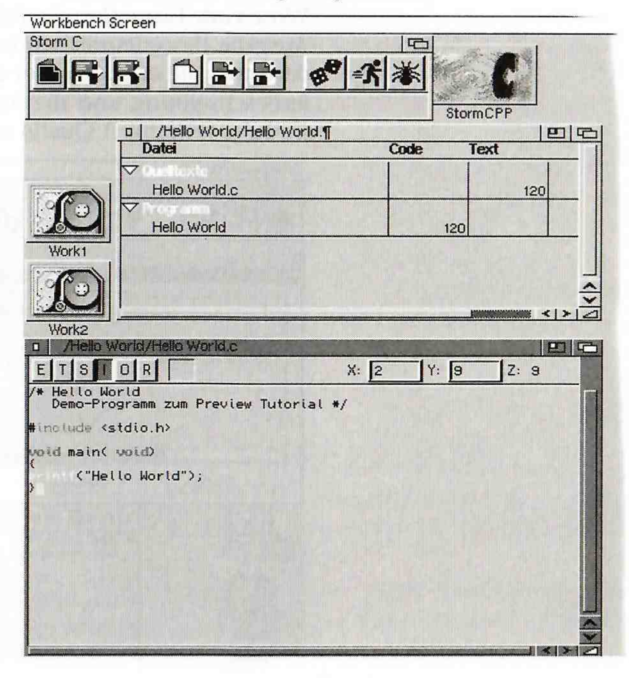
Speichern Sie jetzt bitte das eingegebe Programm unter dem Namen `"Hello World.c"` im Verzeichnis `"Hello World"`.

Rufen Sie dann den Menüpunkt ***„Projekt/Fenster hinzufügen"*** auf. Der Dateiname erscheint nun in der Quelltext-Sektion des Projektfensters. Anhand der Dateiendung entscheidet die Projektverwaltung, in welche Sektion die Datei abgelegt wird.

Bevor Sie nun den Compiler starten, sollten Sie noch einen Dateinamen für das Programm angeben. Andernfalls erzeugt der Linker automatisch den Dateinamen `"A.Out"`.

Wählen Sie dazu aus dem Menü ***„Projekt"*** den Eintrag ***„Programmname wählen"*** aus. Achten Sie bitte darauf, daß das Projektfenster das aktive Fenster ist.

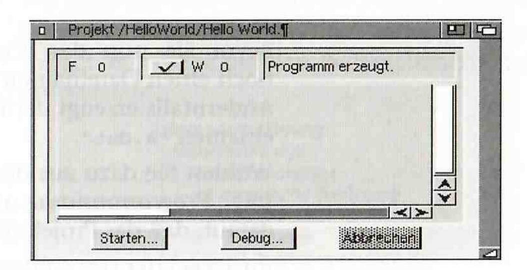
Ihr Projektfenster sollte jetzt so aussehen, wie in der nächsten Abbildung dargestellt.



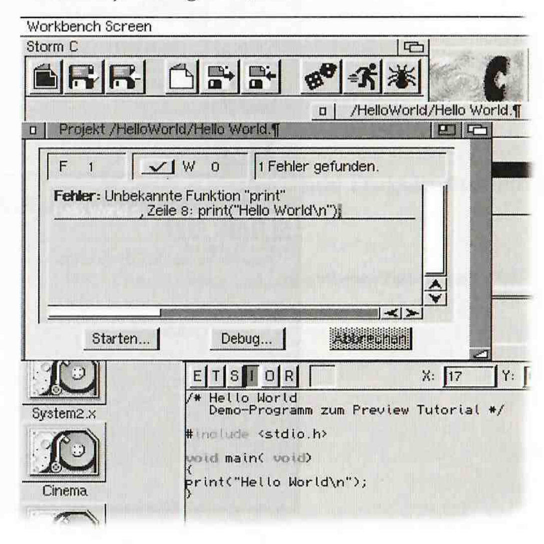
Quelltext kompilieren

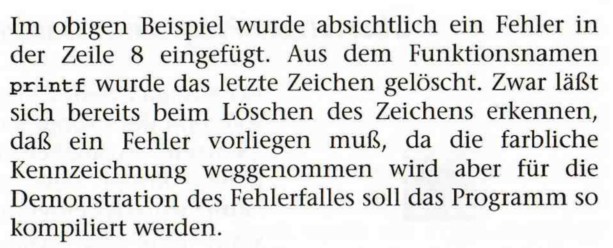


Mit Klick auf das Projekt-Piktogramm **„Make"** wird das Compiler-Fehlerfenster geöffnet und der Quelltext vom Compiler übersetzt. Während der Übersetzung werden vom Compiler Statusmeldungen im Fehlerfenster ausgegeben.



Wird vom Compiler ein Fehler entdeckt, erfolgt die Ausgabe des entsprechenden Fehlers in der Anzeige darunter. Sie erhalten hier eine sehr detaillierte Fehlerbeschreibung und die Angabe der Zeilennummer und des jeweiligen Quellsources.





Im obigen Beispiel wurde absichtlich ein Fehler in der Zeile 8 eingefügt. Aus dem Funktionsnamen `printf` wurde das letzte Zeichen gelöscht. Zwar läßt sich bereits beim Löschen des Zeichens erkennen, daß ein Fehler vorliegen muß, da die farbliche Kennzeichnung weggenommen wird aber für die Demonstration des Fehlerfalles soll das Programm so kompiliert werden.

Sobald der Compiler auf den Fehler trifft, wird eine entsprechende Fehlermeldung im Fenster ausgegeben.

Selbstverständlich genügt ein Doppelklick auf den jeweiligen Fehler und die Projektverwaltung lädt den Quelltext und zeigt die Fehlerstelle im Editor-Fenster an. Sie können dann die Korrektur vornehmen und den Compilerlauf erneut starten.

Übersetztes Programm starten



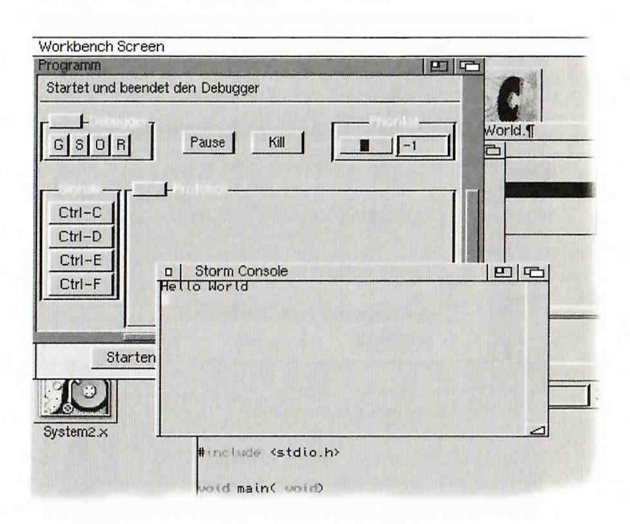
Bei erfolgreichem Compilieren und Linken wird der während des Compilierens nicht zugängliche Knopf „**Starten**" zugänglich und anklickbar. Mit Klick auf diesen Knopf oder auf das Projekt-Piktogramm "**Run**" wird das Programm gestartet.

Wird statt des „**Make**"- das „**Run**"-Piktogramm angeklickt, prüft die Projektverwaltung zuerst, ob bereits alle Module kompiliert sind. Ist dies nicht der Fall, wird automatisch der Compiler für alle noch nicht übersetzten Projektmodule gestartet. Nach der erfolgreichen Übersetzung, wird dann das Programm automatisch gestartet.

Die RunShell

Einen Programmstart aus der Projektverwaltung ist ein sehr komplexer Vorgang, auf den ich hier näher eingehen möchte.

Wie Sie sicher bemerken, wird beim Starten ein weiteres Fenster geöffnet, die sogenannte „RunShell".





Selbstverständlich ist es auch möglich, das Programm einfach so zu starten, aber in einer Entwicklungsumgebung wünscht man sich ja, daß das Programm auch nach dem Starten noch beeinflußt und debuggt werden kann. So ist es z. B. möglich, auch noch nach dem Starten des Programmes den Debugger aufzurufen, um das Programm genauer unter die Lupe zu nehmen. Auch wenn ein Programmfehler auftritt, der eine CPU-Exception auslöst, wird dieser abgefangen und in den meisten Fällen die Möglichkeit geboten, die exakte Fehlerstelle im Quellcode zu begutachten.

Eine weitere wichtige Eigenschaft ist das sogenannte „Resourcen Tracking", das von der RunShell vorgenommen wird. Hierbei werden die Systemfunktionen dokumentiert, die mit dem „Resourcen-Handling" zu tun haben (AllocMem, OpenWindow, Open, ...). Beim Beenden des Programmes kann so genau festgestellt werden, welche Ressource nicht oder welche zuviel freigegeben wurde. Selbstverständlich wird auch hierbei direkt die Quelltextstelle angezeigt und der Fehler kann gleich geändert werden.

Desweiteren bietet die RunShell die Möglichkeit, die Signale ***„Ctrl-C, Ctrl-D, Ctrl-E, Ctrl-F"***, die normalerweise nur per Shell (CLI), dem von eben dieser aus gestarteten Programm übermittelt werden können.

Auch die Priorität, mit der das System das Programm bearbeiten soll, kann in Schritten zwischen -128 bis +127 eingestellt werden.

Der Knopf **„Pause"** hält das Programm in dem gerade aktiven Zustand an. ***„Pause"*** ist ein Toggle-Knopf, das heißt er wechselt bei jedem Klicken den Zustand. Ein Klick aktiviert den Knopf und das Programm wird angehalten. Ein weitere Klick und das Programm läuft wieder weiter. Der Knopf wird dann wieder inaktiv angezeigt.

Mit Klick auf den Knopf **„Kill"** wird das Programm gleich beendet. Alle aufgezeichneten Ressourcen werden zuvor freigegeben (Speicher und Signale werden freigegeben, Screens, Fenster, Requester und

Dateien geschlossen). Auf diese Art und Weise sind Programmleichen ausgeschlossen. Durch versehentlich programmierte Endlos-Schleifen entstehen leicht solche Programmleichen. Die Methode ist wesentlich zeitsparender und damit effizienter als den Amiga neu zu starten.

In unserem Beispiel wird das Programm allerdings so schnell abgearbeitet, so daß das Fenster der RunShell nur sehr kurze Zeit auf dem Bildschirm zu sehen ist. Im nächsten Beispiel werden Sie länger mit der RunShell konfrontiert, da Sie hierbei mit dem Debugger arbeiten werden.

Bei dem Ausgabefenster, das immer noch den Inhalt `"Hello World"` anzeigt, handelt es sich um ein normales Console-Fenster, das auch nach dem Programmende noch geöffnet bleibt. Console-Fenster werden automatisch geöffnet, wenn vom Programm Ausgaben auf die Standard-Ausgabe wie z. B. mit der Funktion `printf` gemacht werden. Klicken Sie einfach auf das Schließfeld des Fensters und es wird geschlossen.

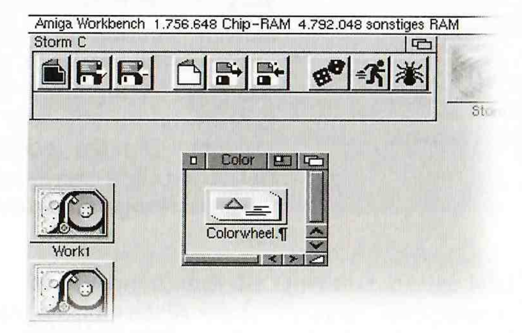
Debugger

Ein Debugger wird zur schnellen Fehlersuche benötigt. Er bietet die Möglichkeit, gezielt Unterbrechungspunkte in Ihren Programmen zu definieren und an diesen „Breakpoints" Veränderungen von Variablen, Strukturen und Klassen zu beobachten. Auf diese Weise lassen sich Fehler einkreisen und können schnell behoben werden.

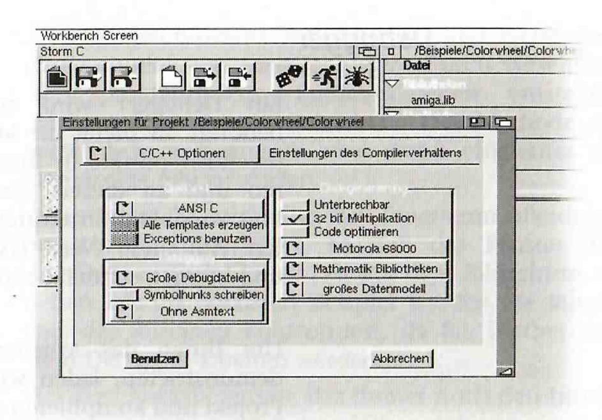
Um Ihnen das Arbeiten mit dem Debugger zu demonstrieren, laden wir ein bereits vorgefertigtes Projekt und kompilieren es.



Im Verzeichnis Beispiele finden Sie den Ordner Colorwheel. Bitte öffnen Sie den Ordner auf der Workbench. Das Bild auf Ihrem Monitor sollte so aussehen, wie in der unteren Abbildung dargestellt.

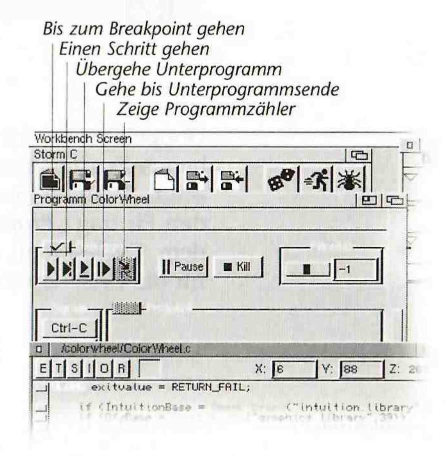


Doppelklicken Sie auf das Projekt-Ikon im Workbench-Fenster. Das Projekt wird geladen und angezeigt. Wählen Sie bitte in ***„Menü-Einstellungen"*** den Eintrag ***„Projekt..."*** aus. Wechseln Sie dann mit dem Cycle-Menü oben links zu den Einstellungen für die C/C++ Optionen.



Wie Sie erkennen können, ist bereits voreingestellt, daß für alle Module große Debugger-Dateien (inkl. aller Include-Dateien) gespeichert werden sollen.

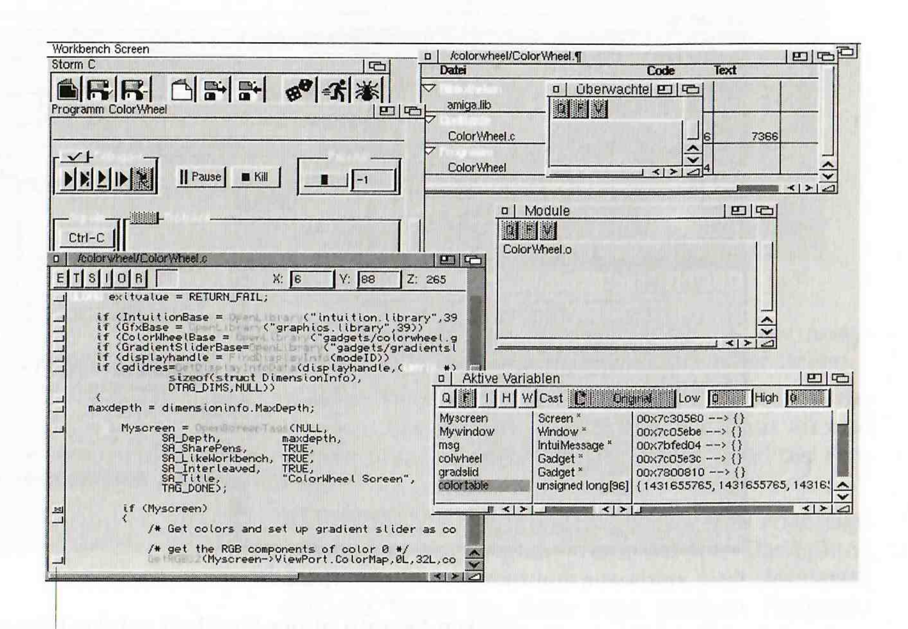
Starten Sie nun den Compiler mit Klick auf das Debugger-Piktogramm. Genau wie beim Klick auf das Run-Piktogramm, wird auch hierbei zuerst geprüft, ob für alle oder für einzelne Module bereits Debugger-Dateien erstellt wurden, und eventuell neu kompiliert werden müssen. Nach dem Linken wird das Programm direkt im Debug-Modus gestartet.





Je nach Voreinstellung wird das Modul-, Aktive-Variablen- und Überwachte-Variablen-Fenster geöffnet.

Desweiteren wird der Quelltext des Moduls, das die `main`-Funktion enthält, geöffnet und der Quelltext ab der `main`-Funktion angezeigt.



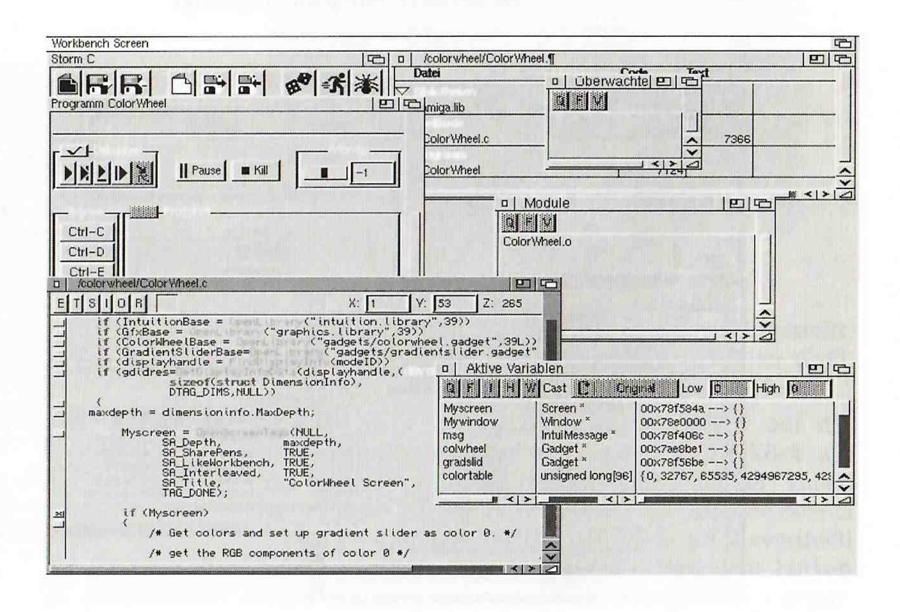
Spalte für Breakpoints

Sie erkennen, daß die Quelltextanzeige leicht verändert dargestellt wird. Die erste Textspalte verschiebt sich dabei nach rechts, so daß eine weitere Spalte für die Unterbrechungspunkte angezeigt werden kann. Jeder Unterbrechungspunkt in der Spalte steht vor einer Quelltextzeile, an der das Programm angehalten werden kann. Wenn Sie mit der Maus auf einen dieser Punkte klicken, wird dieser mit einem diagonalen Kreuz markiert und ist damit ein aktiver Unterbrechungspunkt.

Bitte setzen Sie einen Unterbrechungspunkt direkt nach dem OpenScreen-Aufruf.

Sollte das Fenster der aktuellen Variablen nicht geöffnet sein, öffnen Sie es bitte durch Auswählen der Menüleiste ***„Fenster/Aktuelle Variablen"***.

Ihr Bildschirm sollte nun so aussehen, wie in der nächsten Abbildung dargestellt.

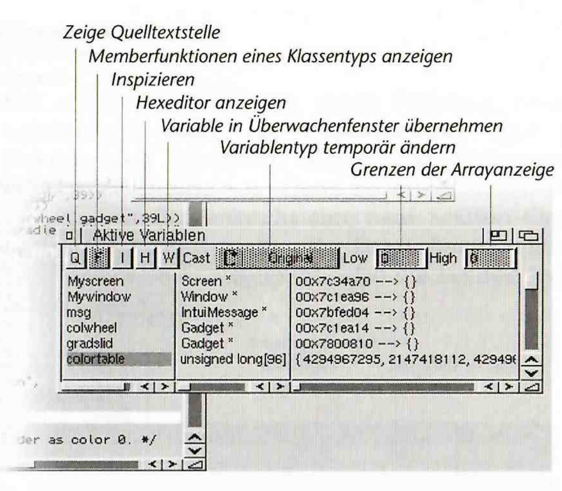


Klicken Sie nun in der RunShell auf das Piktogramm ***„Gehe bis zum nächsten Breakpoint"***.

Da Sie den Unterbrechungspunkt unmittelbar nach dem Funktionsaufruf `OpenScreen` gesetzt haben, wird das Programm bis dorthin ausgeführt und ein neutraler Screen geöffnet. Die Programmausführung ist nun wieder gestoppt.

Mit dem Ausführen der nächsten Funktion `GetRGB32` wird ein Array von `unsigned long`-Typen mit Daten versorgt, was wir uns etwas genauer ansehen wollen.

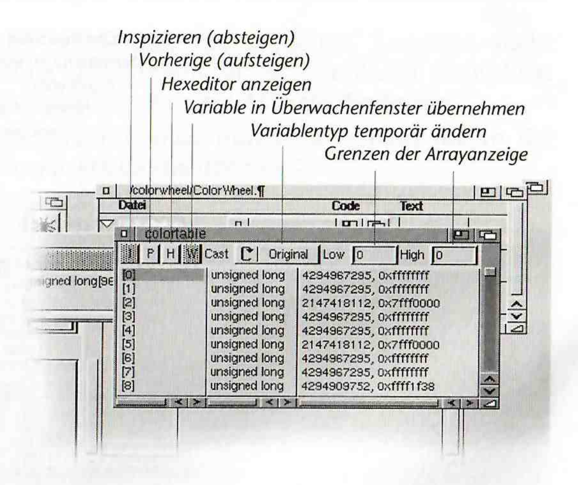




Dazu ist es zunächst notwendig, daß Sie die Array-Variable `colortable` in ein Inspektorfenster stellen.

Wählen Sie einfach im Fenster der aktuellen Variablen die Variable `colortable` mit der Maus an und klicken auf das Symbol "I" am oberen Rand des Fensters.

Das nachfolgend abgebildete Fenster wird angezeigt. Sollten die angezeigten Werte bei der Darstellung auf Ihrem Monitor anders aussehen, liegt das daran, daß die Werte im Array zum jetzigen Zeitpunkt noch undefiniert sind. Das heißt, daß dort der gerade aktuelle Speicherinhalt angezeigt wird und dieser kann bei Ihnen andere Werte enthalten.



Als nächstes führen Sie drei Einzelschritte aus und beobachten dabei genau, wie sich der Inhalt des Inspektorfensters verändert. Klicken Sie dazu zweimal auf das Symbol ***„Einen Schritt gehen"*** in der RunShell. Die Funktion `GetRGB32` lädt die in der Screen View-Struktur vorgegebenen Werte in das Array `colortable`.

Anhand des Beispiels sehen Sie, wie einfach es ist, Variableninhalte mit dem Inspektorfenster anzeigen zu lassen.

Setzen Sie nun weitere Unterbrechungspunkte und spielen Sie etwas mit den Debugger-Funktion, um damit vertraut zu werden. Sie werden sehr schnell feststellen, wie einfach das System zu bedienen ist.

Um den Debugger zu beenden, genügt es, das im Debugger-Modus gestartete Programm zu beenden.

Es werden automatisch alle Debugger-Fenster geschlossen und die RunShell beendet.



Projektsektionen

Die einzelnen Sektionen eines Projektes werden automatisch durch die Dateiendungen und bei Dokumenten zusätzlich durch die Dateinamen erzeugt. Wird eine ".c"-Datei zu einem neuen Projekt hinzugefügt, entsteht eine neue Sektion Quelltexte mit der entsprechenden Datei. Beim weiteren Hinzufügen von Quelltexten wird die Sektion lediglich erweitert.

Sektionenbereich

Codegröße der jeweiligen Objektdateien

Textgröße in Byte

Faltenschalter - Bei Mausklick öffnet oder schließt sich die Falte.

Ram Disk:Demo.¶

Datei	Code	Text
amiga.lib		
Quelltexte		
Demo.c		
Demo.cc		
Demo.ccp		
Demo.c++		
Demo.cpp		
Demo.h		
Demo.hh		
Demo.hhp		
Demo.h++		
Demo.hpp		
Demo.asm		
Demo.ass		
Demo.s		
Demo.i		
Demo.dok		
Demo.doc		
Demo.txt		
Demo.readme		
lies mich		
liesmich		
readme		
read me		
read.me		
lies.mich		
Demo.rexx		
Irgendwas		
Test.¶		
Test.guide		
Demo		

Eigenheiten des Compilersystems

Trotz der ANSI-Spezifikationen hat jedes Compilersystem so seine Eigenheiten. Eingeleitet werden diese Compilerspezifischen Dinge mit einem `#pragma`.

`#pragma`-Zeilen werden, genau wie `#include`, vom Preprozessor interpretiert und ausgeführt. Mit einem `#pragma` realisiert man dabei compilerspezifische Funktionen, die ausdrücklich nicht standardisiert sind.

Compilermodus

`#pragma -` ist ein nicht standardisiertes Feature und schaltet in StormC den Compiler um auf den ANSI-C Modus.

`#pragma +` bewirkt das der Compiler den Quelltext im C++-Modus übersetzt.

Daten in Chip- und Fast-Ram

Die Architektur des Amiga ist etwas unkonventionell. So gibt zwei, manchmal auch mehr verschiedene Klassen von RAM.

In der Praxis interessiert den Programmierer aber nur die Frage „Chip- oder Fast-RAM", denn Sie müssen dafür sorgen, daß Grafikdaten im Chip-Memory abgelegt werden. In StormC gibt es dafür die Pragmas `"chip"` und `"fast"`.

Alle nach der Zeile

```
#pragma chip
```

deklarierten statischen Daten werden ins Chip-RAM geladen,

```
#pragma fast
```

schaltet wieder in den normalen Modus, in dem die Daten irgendwo abgelegt werden, bevorzugt aber im Fast-RAM.



Breakpoint-Kontrolle

Die Breakpoint-Kontrolle hat nichts mit den Breakpoints im Debugger zu tun sondern vielmehr mit der Compilermöglichkeit, Programme zu erzeugen, die vom Benutzer abgebrochen werden können. Das ist einerseits während der Programmentwicklung angenehm, wenn man versehentlich eine Endlosschleife programmiert hat, und andererseits ist es oft sinnvoll, wenn der Anwender eines Programms die Programmausführung gewaltsam abbrechen kann.

Ein Programm fängt einen Abbruch ab, indem es an bestimmten Stellen, z.B. mindestens einmal innerhalb jeder Schleife, das Signalbit prüft, das durch die Tastenkombination <CTRL>+<C> oder das CLI-Kommando `"break"` gesetzt wird. Diese Test kosten selbstverständlich sehr viel Rechenzeit. Außerdem ist es oft gar nicht wünschenswert, wenn eine bestimmte Aktion einfach so abgebrochen wird. Deshalb ist die Abbruchmöglichkeit optional. Im Einstellungen-Dialog gibt es dafür den Schalten **„Unterbrechbar"**.

Sollten Sie jedoch in einigen Programmteilen die Abbruchmöglichkeit benutzen wollen und in anderen nicht, gibt es dafür das Pragma `"break"`.

```
#pragma break +
```

schaltet das setzen der Unterbrechungspunkte ein und

```
#pragma break -
```

wieder aus.

Drück man bei angeschalteter Abbruchmöglichkeit <CTRL>+<C>, entspricht dies einem `"exit(900)"`. Es werden alle Dateien geschlossen und Speicher und sonstige Ressourcen freigegeben. Auch mit `"atexit"` eingehängte Funktionen werden noch ausgeführt. Natürlich ist ein solcher Abbruch ein Low-Level-Konstrukt, so daß dabei keine Destruktoren aufgerufen werden.

Betriebssystemaufrufe

Die Amiga Betriebssystem-Funktionen werden mit einem `#pragma amicall` aufgerufen. Eine solche Deklaration besteht im Wesentlichen aus vier Teilen:

- Dem Namen der Basisvariablen.
- Dem Offset als positive Ganzzahl.
- Dem Funktionsnamen, der bereits deklariert sein muß. Aus Gründen der Eindeutigkeit darf dieser Funktionsname nicht überladen sein.
- Der Parameterliste, vertreten durch eine entsprechende Anzahl von Registernamen in runden Klammern.

Ein Beispiel:

```
#pragma amicall(Sysbase, 0x11a, AddTask(a1,a2,a3))
#pragma amicall(Sysbase, 0x120, RemTask(a1))
#pragma amicall(Sysbase, 0x126, FindTask(a1))
```

Solche Deklarationen werden Sie normalerweise nie selbst schreiben müssen, da alles mit den Amiga-Libraries mitgeliefert wird.

Zeilen verbinden

In C wie in C++ ist es absolut egal wo einzelne Zeilen enden. Beim Preprozessor dagegen ist es sehrwohl relevant, denn hier muß jede Anweisung genau eine Zeile umfassen. Natürlich kann es dazu kommen, daß eine Zeile, z.B. durch eine umfangreiche Makrodefinition, sehr lang und unhandlich wird. Für diesen Fall gibt es den Backslash (Gegenschrägstrich). Wenn ein "\" am Zeilenende steht, wird dies Zeile mit der nachfolgenden verbunden, z.B. so:

```
#define IRGENDWAS \
47081115
```

Dies ist eine gültige Makrodefinition, denn "47081115" wird hier in die vorhergehende Zeile gezogen.



Vordefinierte Symbole

Der Preprozessor besitzt viele vordefinierte Makros. Dabei handelt es sich zum Teil um ANSI C genormte, andere sind Bestandteil von C++ oder spezielle Eigenheiten von StormC. Diese Makros können nicht umdefiniert werden.

`__COMPMODE__`

ist in StormC definiert mit der "`int`"-Konstante `0` im C-Modus und unter C++ mit `1`.

`__cplusplus`

In StormC ist das Makro "`__STDC__`" sowohl im C- als auch im C++-Modus definiert. Will manin StormC nun explizit abprüfen, ob im C++-Modus kompiliert wird, muß dies mit dem Makro "`__cplusplus`" gemacht werden.

`__DATE__`

Das Makro __DATE__ liefert das Datum der Übersetzung. Dies ist sehr nützlich, wenn man ein Programm mit einer eindeutigen Versionsnummer versehen will:

```
#include <stream.h>
void main()
{ cout « "Version 1.1 vom " __DATE__", "__TIME__" Uhr\n; }
```

Das Datum wird dabei in der Form „Monat Tag Jahr", z.B. "Feb 08 1996", geliefert, während die Uhrzeit ganz normal in der Form „Stunde:Minute:Sekunde" vorliegt.

`__FILE__`

Dieses Makro enthält den Namen der aktuellen Quelltextdatei in Form eines Strings, z.B. so:

```
#include <stream.h>

void main()
{ cout « "Das steht in Zeile " « __LINE__ « " in der Datei
" __FILE__ ".\n";}
```

Der Wert des Makros `__FILE__` ist eine konstante Zeichenkette und kann als solche natürlich auch mit davor oder dahinter stehenden Zeichenketten verbunden werden.

`__LINE__`

Das Makro __LINE__ liefert die Zeilennummer, in der es benutzt wird, als dezimale "int"-Konstante.

`__STDC__`

Dieses Makro liefert in alles zu ANSI C kompatiblen Implementationen den numerischen Wert 1 und soll ermöglichen, festzustellen, ob ein Compiler diesem Standard genügt, denn anderenfalls ist `"__STDC__"` nicht definiert.

`__STORMC__`

Oft möchte man gerne testen, um welchen Compiler und um welche Compilerversion es sich handelt. Deshalb definiert StormC zusätzlich ein Makro mit dem Namen `__STORMC__`.

`__TIME__`

(siehe `__DATE__`)

Menübefehle

Projekt

Neu   **A-N**

Bei aktivem Piktogramm- oder Projektfenster wird ein neues Projekt geöffnet und entspricht damit einem Klick auf das Piktogramm **„Neues Projekt"**.

Wenn ein Editorfenster bei der Auswahl des Menüpunktes aktiv ist, wird ein neues Editfenster geöffnet. Das entspricht einem Klick auf das Piktogramm **„Neuer Text"**.

Öffnen...   **A-O**

Signalisiert durch die drei Punkte am Ende des Menüeintrags ***„Öffnen..."*** erscheint ein weiterer Dialog, der auf Anwenderreaktionen wartet, bevor die Aktion ausgeführt wird. Im Fall von ***„Öffnen..."*** wird der Standard-Dateiauswahldialog (ASL-Library) zum Einlesen von Dateien angezeigt. Die Bedienung der Dialogbox sollte Ihnen bekannt sein, da sie in sehr vielen Programmen vorkommt.

Je nach dem, ob das Piktogramm-/Projektfenster oder ein Editorfenster aktiv ist, kann entweder ein Projekt oder ein Text geladen werden. Die Piktogrammleiste bietet ebenfalls zwei Piktogramme zum Laden von Quelltexten und Projekten.

Sichern   **A-S**

Bei aktivem Projektfenster wird das Projekt gespeichert und entspricht damit einem Klick auf das Piktogramm **„Projekt speichern"**.

Wenn ein Editorfenster bei der Auswahl des Menüpunktes aktiv ist, wird der Text gespeichert. Das entspricht einem Klick auf das Piktogramm **„Text speichern"**.

Sichern als...

Genauso wie bei der Auswahl von ***„Öffnen..."*** wird auch bei ***„Speichern als..."*** der Dateiauswahldialog angezeigt.

Sie haben die Möglichkeit, einen Dateinamen für Ihr Projekt oder Ihren Text anzugeben. Die Auswahl per Mausklick in die angezeigte Dateiliste ist ebenfalls möglich.

Je nachdem, ob das Projektfenster oder ein Editorfenster aktiv ist, kann entweder das Projekt oder der Text gespeichert werden. Die Piktogrammleiste bietet ebenfalls zwei Piktogramme zum Speichern von Quelltexten und Projekten.

Sichern als Projektvorlage

Dieser Menüpunkt ist nur bei aktivem Projektfenster anwählbar. Die Projektvorlage ist die Datei, die geladen wird, wenn Sie ein neues Projekt entweder über das Menü ***„Projekt/Neu"*** oder über das Piktogramm **„Neues Projekt"** anlegen. Sie haben damit die Möglichkeit, eigene Einstellungen vorzugeben, die dann bei jedem Ihrer nächsten Projekte verwendet werden.

Gespeichert werden alle Einstellungen, die Sie zum Projekt machen können (C/C^{++} Umgebung, C/C^{++} Preprozessor, C/C^{++} Optionen, C/C^{++} Warnungen, Linker Optionen und Programmstart) und selbstverständlich auch alle, in dem zum Speichern vorbereiteten Projekt enthaltene Sektionen.

Alles sichern

Mit Klick auf den Menüpunkt ***„Alles sichern"*** werden alle angezeigten und noch nicht gesicherten Quelltexte und Projekte gespeichert. Sind für Quelltexte oder Projekte noch keine Dateinamen angegeben, wird für jeden fehlenden Dateinamen die Dateiauswahl geöffnet.

Datei hinzufügen...

Dieser Menüpunkt ist nur anwählbar, wenn ein Projektfenster aktiviert ist. Aus der angezeigten Dateiauswahl kann eine Datei ausgewählt werden, die dann in das Projekt aufgenommen wird. Je nach Dateiendung werden verschiedene Sektionen im Projekt erzeugt und die Datei an die entsprechende Stelle im Projekt plaziert.

Fenster hinzufügen

Bei aktiviertem Editorfenster und angelegtem Projekt wird dieser Menüpunkt zugänglich. Mit Klick auf ***„Fenster hinzufügen"*** wird die Datei im aktiven Editorfenster in die entsprechende Projektsektion aufgenommen. Man spart sich dadurch den Umweg über die Dateiauswahl, die erscheint, wenn ***„Datei hinzufügen"*** gewählt wird.

Programmname wählen...

Selbstverständlich benötigt jedes, aus mehreren Modulen erzeugte Programm einen Namen. Da der Programmname nicht anhand der Dateiendung ermittelt und somit automatisch in die richtige Sektion eingetragen werden kann, muß man ihn manuell angeben. Dazu wählen Sie einfach den Menüeintrag ***„Programmname wählen..."*** und geben in der angezeigten Dateiauswahl den entsprechenden Dateinamen an. Sie haben damit auch die Möglichkeit, das aus den Modulen erzeugte Programm an einer anderen Stelle auf Ihrer Festplatte speichern zu lassen, als im Projektverzeichnis.

Schließen A-K

Je nachdem, ob ein Projekt- oder Editorfenster aktiv ist, wird entweder das Projekt oder der Text geschlossen. War das Projekt oder der Text zuvor noch gespeichert, erscheint eine Sicherheitsabfrage und bietet Ihnen die Möglichkeit, dieses noch zu tun.

Das gleiche kann mit Klick auf das Schließfeld des jeweiligen Fensters erreicht werden.

Über

In dem angezeigten Dialog finden Sie Informationen zum Produkt-Copyright, unsere aktuelle Telefon-, Faxnummer und eine E-Mail-Adresse.

Beenden A-Q

Ein Klick auf diesen Menüpunkt beendet das Entwicklungssystem. Sind noch Quelltexte oder Projekte angezeigt, die noch gespeichert werden müssen, erhalten Sie einen entsprechenden Hinweis und die Möglichkeit, dies zu erledigen.

Bearbeiten

Markieren A-B

Der Menüpunkt ist nur bei aktiviertem Editorfenster wählbar und schaltet den Editor-Markierungsmodus an oder ab.

Ausschneiden A-X

Der Menüpunkt ist nur bei aktiviertem Editorfenster wählbar. Der markierte Textbereich wird dabei in die Zwischenablage kopiert und aus dem aktiven Text gelöscht. Die Markierung wird ebenfalls aufgehoben.

Kopieren A-C

Der Menüpunkt ist nur bei aktiviertem Editorfenster wählbar. Anders als beim Ausschneiden wird mit Kopieren der markierte Textbereich lediglich in die Zwischenablage kopiert aber nicht aus dem aktiven Text gelöscht.

Einfügen A-V

Der Menüpunkt ist nur bei aktiviertem Editorfenster wählbar. Dabei wird der in der Zwischenablage befindliche Text ab der Stelle des Textcursors in den aktiven Text eingefügt.



Löschen

Bei aktivem Editorfenster wird der markierte Textbereich aus dem Text entfernt. Der gelöschte Text wird nicht in die Zwischenablage kopiert. Das Textlöschen kann jedoch mit *„Undo"* rückgängig gemacht werden.

Bei aktivem Projektfenster wird das in einer Sektion markierte Modul aus dem Projekt entfernt. Ein *„Undo"* ist hierbei nicht möglich.

Undo A-Z

Der Menüpunkt ist nur bei aktiviertem Editorfenster wählbar. Mit *„Undo"* wird die zuletzt ausgeführte Editorfunktion zurückgenommen.

Redo A-R

Der Menüpunkt ist nur bei aktiviertem Editorfenster wählbar. Mit *„Redo"* wird das zuletzt ausgeführte *„Undo"* zurückgenommen.

Finden & Ersetzen... A-F

Finden & ersetzen ist selbstverständlich nur bei aktivem Editor-Fenster wählbar. Wie die drei Punkte nach dem Menüeintrag symbolisieren, wird hierbei ein Dialog angezeigt, in dem der Suchbegriff und das Wort, durch das ersetzt werden soll, eingegeben werden können.



Die Funktion kann natürlich auch nur zum Suchen verwendet werden. Lassen Sie hierzu einfach den Ersetzen-Begriff frei und klicken auf das Symbol **„Finden"**.

Zusätzlich kann die Suchrichtung ab der Cursorposition angegeben werden. Es besteht aber auch die Möglichkeit immer den kompletten Text durchsuchen zu lassen, egal an welcher Stelle gerade der Textcursor positioniert ist.

Mit der Auswahl einer Spezifiktaion aus dem Cycle-Menü darunter lassen sich die Suchparameter exakter definieren.

Mit den rechts daneben angebrachten Symbolen zum Abhaken, kann eingestellt werden, daß die Groß-/Kleinschreibung und eventuell enthaltene Akzente nicht beachtet werden sollen.

Mit den drei Symbolen am unteren Rand des Dialogs führen Sie die gemachten Anweisungen auf verschiedene Weise aus.

„Finden" sucht lediglich den im Finde-Textsymbol eingegebenen Begriff.

„Ersetzen" ersetzt den Finde-Begriff durch den Ersetze-Begriff einmalig.

„Alle ersetzen" ersetzt alle gefunden Finde-Begriffe durch den Ersetze-Begriff.

Finde nächstes A- "."

Das Menü kann nur bei aktivem Editor-Fenster ausgewählt werden.

„Finde nächstes" führt die zuletzt gemachte Finde - Anweisung aus, ohne zuvor den Dialog zu öffnen.

Ersetze nächstes A- "-"

Das Menü kann nur bei aktivem Editor-Fenster ausgewählt werden.

„Ersetze nächstes" führt die zuletzt gemachte Ersetze-Anweisung aus, ohne zuvor den Dialog zu öffnen.



Kompilieren

Kompilieren...

Der Menüpunkt ***„Kompilieren..."*** ist anwählbar, wenn in der Projektsektion Quelltexte ein Eintrag markiert ist. Er kann auch dann angewählt werden, wenn der im aktiven Quelltexteditor-Fenster angezeigte Quelltext im aktiven Projekt enthalten ist.

Mit dieser Funktion können Sie einzelne Module gezielt kompilieren lassen.

Make... A-M

Der Menüpunkt ist nur bei aktiviertem Projekt- oder Fehlerfenster wählbar. Er kann auch dann angewählt werden, wenn der im aktiven Quelltexteditor-Fenster angezeigte Quelltext im aktiven Projekt enthalten ist.

Mit Make werden alle Module kompiliert, die seit dem letzten Kompilieren verändert wurden. Dabei werden auch Abhängigkeiten zwischen Programm- und Header-Dateien berücksichtigt.

Ein Klick auf das Piktogramm **„Make"** hat dieselbe Auswirkung.

Starten...

Der Menüpunkt ist nur bei aktiviertem Projekt- oder Fehlerfenster wählbar. Er kann auch dann angewählt werden, wenn der im aktiven Quelltexteditor-Fenster angezeigte Quelltext im aktiven Projekt enthalten ist.

Falls für das aktuelle Projekt bereits ein ausführbares Programm erzeugt wurde, wird dieses gestartet. Zuvor wird ein **„Make"** ausgeführt und damit automatisch alle veränderten Module neu kompiliert und auch das Programm neu erzeugt.

Das selbe kann auch mit Klick auf das Piktogramm **„Run"** erreicht werden.

Debug... A-D

Der Menüpunkt ist nur bei aktiviertem Projekt- oder Fehlerfenster wählbar. Er kann auch dann angewählt werden, wenn der im aktiven Quelltexteditor-Fenster angezeigte Quelltext im aktiven Projekt enthalten ist.

Mit Klick auf ***„Debug..."*** wird praktisch ein ***„Starten..."*** ausgeführt und automatisch der Debugger gestartet. Der Programmzähler wird auf die erste Funktion (`main`) im Programm gesetzt und die Programmausführung angehalten.

Das selbe kann auch mit Klick auf das Piktogramm **„Debug"** erreicht werden.

Berühren (engl. touch)

Der Menüpunkt ist nur bei aktiviertem Projektfenster wählbar.

Mit Klick auf den Menüpunkt ***„Berühren"*** wird der markierte Eintrag in der Quelltext-Sektion als verändert markiert. Bei dem nächsten Starten von ***„Make"*** wird dieser Quelltext garantiert neu kompiliert.

Alles berühren (engl. touch all)

Der Menüpunkt ist nur bei aktiviertem Projektfenster wählbar.

Hierbei werden alle Module in der Quelltext-Sektion als verändert markiert. Bei dem nächsten Starten von ***„Make"*** wird jedes Modul des Projektes neu kompiliert.

Programm sichern als...

Der Menüpunkt ist nur bei aktiviertem Projekt- oder Fehlerfenster anwählbar.

Nach dem erfolgreichen Linken wird das Programm prinzipiell gespeichert. Wenn kein Programmname im Projekt vorgegeben ist, wird das Programm unter dem Namen A.Out in das Verzeichnis gespeichert, in welches das Projekt gespeichert wurde.

Mit dem Menüpunkt „Programm sichern als..." haben Sie nun die Möglichkeit, eine Kopie des



erzeugten Programmes unter anderem Namen an einer anderen Stelle auf Ihrer Festplatte zu speichern.

Fenster

Fehlerfenster...

Der Menüpunkt ist nur bei aktiviertem Projektfenster wählbar.

Das Fehlerausgabefenster wird geöffnet.

Module...

Der Menüpunkt ist nur bei gestartetem Debugger aktiv und öffnet das Fenster der Programmodule, für die Quellinformationen ausgegeben werden können. Prinzipiell handelt es sich dabei um die gleiche Liste von Einträgen, wie sie im Projektfenster unter Quelltexte erscheint, wenn für jeden Quelltext auch eine Debuggerausgabe erzeugt wurde. Es werden lediglich statt der Quelltextenamen die Dateinamen der Objektmodule angezeigt.

Aktuelle Variablen...

Der Menüpunkt ist nur bei gestartetem Debugger aktiv und öffnet das Fenster für die Anzeige der aktuellen Variablen.

Überwachte Variablen...

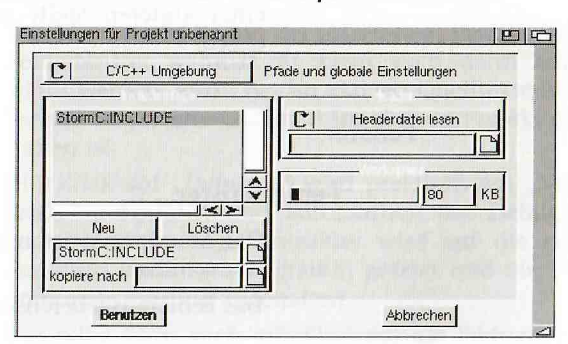
Der Menüpunkt ist nur bei gestartetem Debugger aktiv und öffnet das Fenster für die Anzeige der überwachten Variablen.

Einstellungen

Projekt...

Die Einstellungen zum jeweiligen Projekt werden bei aktiviertem Projekt- oder Fehlerfenster zugänglich. Bei Klick auf das Menü ***„Projekt..."*** wird der abgebildete Dialog angezeigt.

Include-Pfade und vorkompilierte Header



Der erste Einsteller in der Liste der Einstellerdialoge ist der für die Include-Dateien. Aus den Include-Dateien liest der Preprozessor seine Definitionen für z. B. die Standard-Funktionen. In Ihrem Quelltext benutzen Sie zum Einbinden solcher Dateien die Preprozessor-Anweisung `#include`. Vielleicht wissen Sie schon, daß man dabei zwei Möglichkeiten hat: Schließt man den Dateinamen nach dem `#inlcude "so"` in Anführungszeichen ein, sucht der Preprozessor (das Programmodul, das vor dem Compiler die Datei bearbeitet) vom aktuellen Verzeichnis aus, während die `<so>` in spitze Klammern gesetzten Dateinamen Standard-Includedateien bezeichnen und in vorgegebenen Verzeichnissen gesucht werden. In dem Listview können Sie für die Standard-Includedateien mehrere Verzeichnisse vorgeben.

Mit dem **„kopieren nach:"**-Gadget, haben Sie die Möglichkeit einen Kopierpfad für die Dateien anzulegen. Um die Compilergeschwindigkeit zu steigern sollte der hierbei angegebene Pfad in der RAM-Disk liegen, da anderenfalls kein Geschwindigkeitsvorteil zu erwarten ist. Daß Sie über genügend Speicherreserven verfügen sollten, muß ich Ihnen hoffentlich nicht sagen.

Eine weitere Möglichkeit, um Speicher zu sparen und die Compilergeschwindigkeit zu erhöhen, haben Sie durch die Erzeugung von vorkompilierten Header-Dateien. Damit der Compiler weiß, wo die Header aufhören und Ihr eigenes Programm beginnt, muß die Stelle mit einer `"#pragma header"`

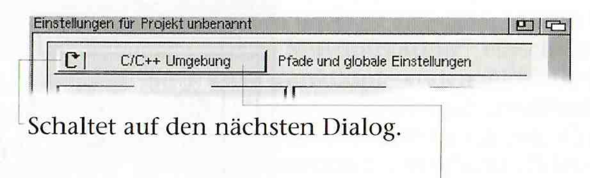


Zeile markiert sein. Am einfachsten schreiben Sie in den Quelltext jedes Moduls zunächst die `"#inclu-des"` aller Header-Dateien, die Sie nicht selbst geschrieben haben oder die sowieso nicht geändert werden, wie z. B. die OS-Includes. Dann folgt die Zeile `"#pragma header"`. Hiernach können Sie dann mit Ihren eigenen Headerdateien und den Rest des Programmes angeben.

Die Pragma-Anweisung hat solange keine Wirkung, bis Sie im Einsteller das Header-Cycle-Menü auf "**Headerdatei schreiben**" einstellen und die veränderten Module neu kompilieren. Sobald der Compiler die Pragma-Anweisung im Programm erreicht, dauert es eine kurze Zeit und die vorkompilierten Header-Dateien werden unter dem angegebenen Namen in das entsprechende Verzeichnis geschrieben.

Vor dem nächsten Compilerlauf schalten Sie dann einfach auf „**Headerdatei lesen**". Der Compiler liest dann die vorkompilierte Headerdatei, sucht nach der Anweisung `"#pragma header"` und setzt dort mit seiner Übersetzung ein.

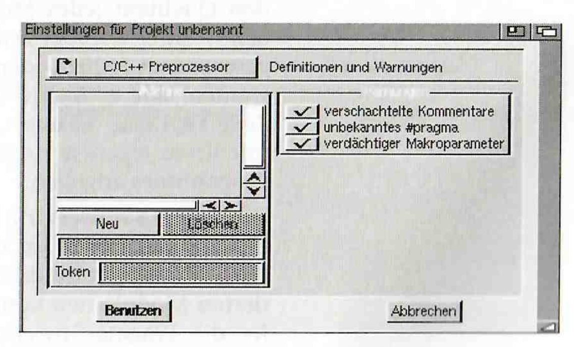
Um auf den nächsten Einsteller wechseln zu können, müssen Sie sich des Cycle-Menüs am oberen Rand des Dialogfensters bedienen.



Schaltet auf den nächsten Dialog.

Zeigt ein Popup-Menü und Sie können aus der Liste der Möglichkeiten wählen.

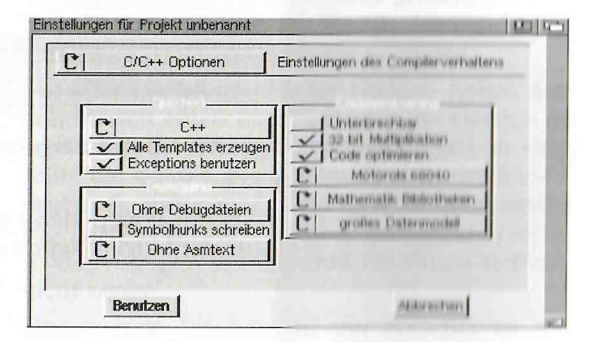
Preprozessor



Mit diesem Dialog können Sie zum einen die Preprozessor-Warnungen einstellen und zum zweiten die Preprozessor-Symbole vordefnieren.

Zu den Warnungen muß ich sicher nichts beschreiben wohl aber zu den Definitionen. Jeder Eintrag, der in das Listview gemacht wird, wirkt beim Compilieren so, als stände er mit einem vorangestellten `"#define"` am Anfang jedes übersetzten Quelltextes. Sie können so global wirkende Definitionen angeben wie z. B. `Debug` und als Token `True` oder `False`.

Codegenerierung und Debugger



Dieser Dialog stellt alle Einstellungen zur Verfügung, die für die Codegenerierung und den Debugger relevant sind. Angefangen mit der Art des Quelltextes (ANSI C oder C++) kann zusätzlich im C++ Modus noch die Verarbeitung von Templates und die



Benutzung des Exception-Handlings eingeschaltet werden.

Unter der Quelltextsteuerung befinden sich die Einsteller für die Debuggerausgabe.

Zuerst wählen Sie mit dem Cycle-Menü, ob überhaupt Debug-Ausgaben erzeugt werden sollen. Der Compiler generiert dann zusätzlich Debug-Dateien mit der Endung `".debug"` und `".link"`. Diese Dateien werden vom Debugger benötigt, um den Bezug zwischen Quelltext und Programmcode herstellen zu können.

Sollten Sie lieber mit einem symbolischen Debugger /Disassembler arbeiten wollen, haben Sie die Möglichkeit, einen **Symbolhunk** an das Programm anhängen zu lassen.

Für diejenigen unter Ihnen, die lieber manuell kontrollieren möchten, was der Compiler so alles generiert, ist die Erzeugung von Assembler-Quelltexten aus den C/C++-Codes möglich. Der Compiler schreibt dann außer der `".o"`-Datei noch eine `".s"`-Datei, die den Assembler-Quelltext wahlweise gemischt mit C-Quelltext enthält.

Mit der Wahl des Symbols **„Codegenerierung unterbrechbar"** erzeugt der Compiler in jeder Schleife eine Abfrage der Tastenkombination <CTRL>+<C>.

Wenn Sie Code für den 68000er erzeugen, sollten Sie den Schalten **„32 Bit Multiplikation"** einschalten, da dann die korrekt funktionierenden Bibliotheksfunktionen zur Langwortdivision und -multiplikation genutzt werden. Bei der Erzeugung von Code für die höheren Prozessoren ist dieser Schalter sowieso bedeutungslos.

Bei eingeschaltetem Schalter **„Code optimieren"**, wird der Programmcode kürzer und meist auch schneller.

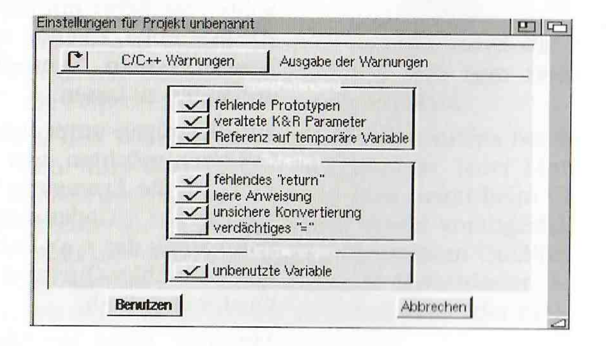
Mit dem Cycle-Menü darunter sind die verschiedenen Prozessoren einstellbar, für die spezieller Code erzeugt werden kann. Hierbei müssen Sie die Abwärts-Inkompatibilität beachten. Wenn Sie zum Beispiel Code für den **68030** erzeugen lassen, ist das Programm nicht mehr auf einem handelsüblichen

Amiga 1200 lauffähig, da dieser mit einem **68020** Prozeossor bestückt ist.

Das Cycle-Menü darunter läßt Ihnen die Wahl zwischen **FPU-Code** oder dem Aufruf der **Systembibliotheken** bei mathematischen Berechnungen in Ihrem Programm.

Das letzte Cycle-Menü in dieser Reihe schaltet zwischen **großem** und **kleinen Datenmodell** um.

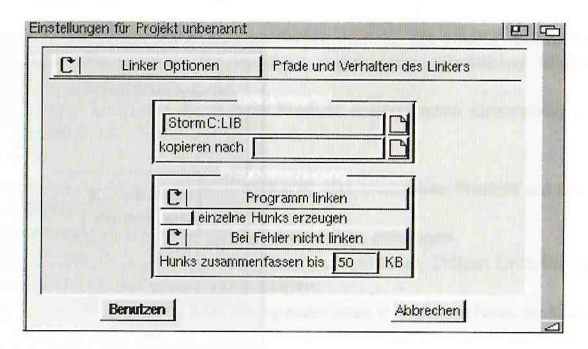
Warnungen



In StormC gibt es 8 Warnungen, die man je nach Situation und persönlichem Programmierstil einzeln ein- und ausschalten kann.



Linker Einstellungen



Ähnlich wie für Include-Dateien bei den Preprozessor-Einstellungen läßt sich auch der Pfad für die Linkerbibliotheken einstellen.

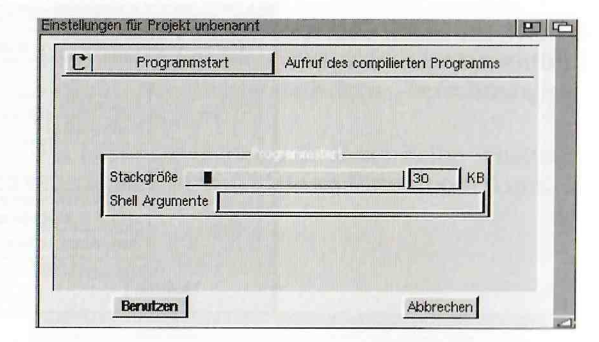
Ist im Cycle-Menü darunter die Einstellung **„Programm linken"** gewählt, wird das übersetzte Programm ganz normal mit den Bibliotheksroutinen zu einem ausführbaren Programm zusammengebunden.

Ist **„Nicht linken"** eingeschaltet, werden keinerlei Bibliotheks- oder sonstige Module hinzugeladen. Der Linker wird überhaupt nicht gestartet.

Bei **„Ohne Statup-Code linken"**, wird das Programm so gelinkt, daß es beispielsweise als „Shared-Library" oder als „Device-Treiber" eingesetzt werden kann. Einen Start solcher Programme von der Workbench oder vom CLI aus ist nicht möglich.

Im Fehlerfall kann mit dem Cycle-Menü darunter festgelegt werden, ob der Linkvorgang gestartet oder nicht gestartet werden soll.

Programmstart



Selbstverständlich muß die Möglichkeit bestehen, für den Start aus der integrierten Umgebung Argumente anzugeben und die Stackgröße des Programmes vorgeben zu können.

Einstellungen laden...

Der Menüpunkt ist nur bei aktivem Debugger anwählbar. Mit dem angezeigten ASL-Dateidialog kann eine Debugger-Settings-Datei mit der Endung `".RUN"` gelanden werden.

Einstellungen sichern

Der Menüpunkt ist nur bei aktivem Debugger anwählbar. Sie können hiermit die komplette Debugger-Konfiguration (welche Fenster geöffnet sind und deren Position) speichern. Gespeichert wird in das Verzeichnis aus dem StromC gestartet wurde unter dem Namen "StormSettings.Run".

Einstellungen sichern als...

Der Menüpunkt ist nur bei aktivem Debugger anwählbar. Es wird die gleiche Konfiguration wie bei „Einstellung sichern" gespeichert mit dem Unterschied, daß der ASL-Dialog erscheint und Sie einen beliebigen Dateinamen und Dateipfad angeben können.



Lizenzvereinbarungen

1 Allgemeines

(1) Gegenstand dieses Vertrages ist das Benutzungsrecht für Computerprogramme der HAAGE & PARTNER Computer GmbH, für die Benutzungsanleitung sowie für sonstiges zugehöriges, schriftliches Material, nachfolgend zusammenfassend als Produkt bezeichnet.

(2) Die HAAGE & PARTNER Computer GmbH und/oder die in dem Produkt angegebenen Lizenzgeber sind Inhaber sämtlicher Rechte an den Produkten und Warenzeichen.

2 Nutzungsrechte

(1) Der Käufer erhält ein nicht übertragbares, nicht ausschließliches Recht, das erworbene Produkt auf einem Computer bzw. an einem Arbeitsplatz zu nutzen.

(2) Darüber hinaus kann der Anwender eine einzige Kopie zu Sicherungszwecken anfertigen.

(3) Der Käufer ist nicht berechtigt, das erworbene Produkt zu vertreiben, zu vermieten, Dritten Unterlizenzen anzubieten oder diese in anderer Weise Dritten zur Verfügung zu stellen.

(4) Es ist verboten, das Produkt zu ändern, zu modifizieren oder anzupassen oder in jeglicher Form rückzuentschlüsseln. Dieses Verbot gilt auch für das Übersetzen, Abwandeln, Rückentschlüsseln und Weiterverwenden von Teilen.

3 Gewährleistung

(1) Die HAAGE & PARTNER Computer GmbH gewährleistet, daß zum Zeitpunkt der Lieferung die Datenträger physikalisch frei von Material- und Herstellungsfehlern sind und das Produkt wie in der Dokumentation beschriebenen Weise genutzt werden kann.

(2) Mängel des gelieferten Produkts werden vom Lieferanten innerhalb der Gewährleistungsfrist von sechs Monaten ab Lieferung nach entsprechender Mitteilung durch den Anwender behoben. Dies geschieht nach Wahl des Lieferanten durch kostenfreie Nachbesserung oder durch Ersatzlieferung in Form eines Updates.

(3) Die HAAGE & PARTNER Computer GmbH übernimmt keine Haftung dafür, daß das Produkt für die vom Kunden vorgesehene Aufgabe geeignet ist. Für eventuell auftretende Folgeschäden übernimmt die HAAGE & PARTNER Computer GmbH keine Haftung.

(4) Der Anwender weiß, daß nach dem heutigen Stand der Technik die Erstellung völlig fehlerfreier Software nicht möglich ist.

4 Sonstiges

(1) In diesem Vertrag sind sämtliche Rechte und Pflichten der Vertragsparteien geregelt. Sonstige Vereinbarungen bestehen nicht. Änderungen sind nur in Schriftform und bei Bezugnahme auf diesen Vertrag wirksam und beiderseitig zu unterzeichnen.

(2) Der Gerichtsstand für alle Streitigkeiten aus diesem Vertrag ist, soweit vereinbar, das zuständige Gericht am Firmensitz der HAAGE & PARTNER Computer GmbH.

(3) Sollten einzelne Bestimmungen dieser Bedingungen nicht rechtswirksam sein oder ihre Rechtswirksamkeit durch einen späteren Umstand verlieren, oder sollte sich in diesen Bedingungen eine Lücke herausstellen, so wird hierdurch die Rechtswirksamkeit der übrigen Bestimmungen nicht berührt. Anstelle der unwirksamen Vertragsbestimmungen oder zur Ausfüllung der Lücke soll eine angemessene Regelung gelten, die, soweit rechtlich möglich, dem am nächsten kommt, was die Vertragsparteien gewollt haben, soweit sie von der Unwirksamkeit der Bestimmung Kenntnis gehabt hätten.

(4) Jede Verletzung vorstehender Lizenzbestimmungen oder von Urheber- und Warenzeichenrechten wird straf- und zivilrechtlich verfolgt.

(5) Durch Installation der Software werden diese Lizenzvereinbarungen anerkannt.

(6) Sind Sie mit den Lizenzvereinbarungen nicht einverstanden, so müssen Sie das Produkt unverzüglich gegen Erstattung bereits geleisteter Zahlungen an den Lieferanten zurückgeben.

Stand: September 1995

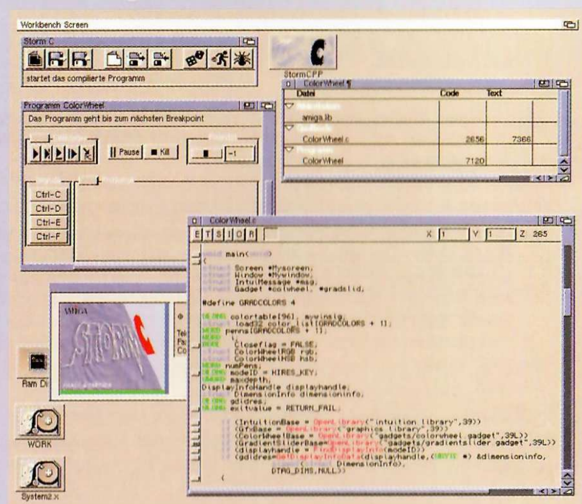
StormC AMIGA-*Entwicklungs-System*

Das vielseitige Programmiersystem für Sie und Ihren Amiga!

StormC beinhaltet alle Entwicklungswerkzeuge, die zum effizienten Entwickeln unerläßlich sind. Dazu gehören:

- *der sehr schnelle* ***Quelltext-Editor****, der Schlüsselworte und Syntaxeigenschaften farbig hervorheben kann. Eine perfekte Quelltext-Strukturierung erhöht die Produktivität.*
- *der Hochgeschwindigkeits* ***ANSI-C/C++ Compiler****. Er generiert Code für alle Prozessoren der Motorola 68000-Familie einschließlich* ***68060****, 68881/68882 und nutzt alle Vorteile dieser schnellen Prozessor-Chips.*
- *die flexible* ***Projektverwaltung****. Mit ihr verwirklichen Sie die komplexesten Aufgaben innerhalb kürzester Zeit.*
- *die* ***RunShell*** *eröffnet vollkommen neue Möglichkeiten bei der Software-Entwicklung auf dem Amiga. Von ihr wird ein „Resource Tracking" durchgeführt, so daß Programmierfehler in einem Stadium entdeckt werden, in dem es bislang keine Lösungen gab.*
- *der besonders komfortable* ***Source-Level-Debugger*** *kann auch während der Laufzeit über die RunShell gestartet werden. Sie müssen sich nicht vor dem Programmstart entscheiden, ob Sie nun debuggen oder das Programm einfach nur testen. Mit der Debugger-Eigenschaft, die Quelltexte im normalen Editor anzeigen zu können, bleiben die Farbmarkierung und Ihre Strukturierung voll erhalten.*
- *die umfangreichen* ***Funktions-Bibliotheken*** *zu ANSI-C, C++ und den Amiga Betriebssystem-Funktionen sind genauso Bestandteil wie die neuen System-Include-Dateien.*
- ***Systemvoraussetzungen:***

  Computertyp:
  alle Amiga-Modelle mit Festplatte
  Betriebssystem:
  AmigaOS 2.0 und höher
  Speicher:
  3 MB Hauptspeicher (einige Funktionen können nur mit min. 6 MB Hauptspeicher genutzt werden)





HAAGE & PARTNER
Computer GmbH
Mainzer Str. 10A
61191 Rosbach v.d.H.

Tel: 0 60 07 / 93 00 50
Fax: 0 60 07 / 75 43
Compuserve: 100654,3133